The Evaluation of Students' Marking in Web-Based Peer Assessment of Learning Computer Programming

Jirarat Sitthiworachart
Department of Computer Science,
University of Warwick, UK
jirarat@dcs.warwick.ac.uk

Mike Joy
Department of Computer Science,
University of Warwick, UK
M.S.Joy@dcs.warwick.ac.uk

Abstract: Peer assessment is a technique that has been successfully employed in a variety of academic disciplines, and it is considered effective in developing students' higher cognitive skills. In this paper, we evaluate students' marking using web-based peer assessment, a novel technology we have introduced to deliver peer assessment in the context of an undergraduate computer programming course. The preliminary results indicate that a web-based peer assessment system can successfully help students to develop their higher cognitive skills in learning computer programming.

Keywords: Peer assessment, Computer programming, Educational software, Evaluation students' performance.

Introduction

Falchikov (2001) defines peer assessment as "the process whereby groups rate their peers". Somervell (1993) states that peer assessment engages students in making judgements on the other students' work. In the peer assessment process, students are involved both in the learning and in the assessment process. Peer assessment is primarily a tool for learning rather than for summative assessment (Brown et al., 1997). Dochy and McDowell (1997) remark "peer assessment is not only a tool to provide a peer with constructive feedback which is understood by the peer. Above all, peer assessment is a tool for the learner himself." The use of peer assessment is claimed to enhance students' evaluative capacities (Fallows & Chandramohan, 2001), which leads to improvements in the quality of their subsequent work (Topping et al., 2000 and Sluijsmans et al., 1999).

Peer assessment is often used in the context of essays, but has seldom been applied to computer programming courses. The skill of writing good software includes an understanding of different approaches to the task, and an appreciation of stylistic and related considerations – these can be developed by evaluating other programmers' solutions. In the peer assessment process discussed in this paper, students become involved in three important activities (i.e. group discussion, marking, and providing feedback), each of which provides benefits to students in improving their learning. When marking, students analyse and evaluate the quality of program, which encourages them to develop higher level cognitive skills in learning computer programming (such as are characterised as 'deep learning' in Bloom's Taxonomy, 1956) as follows:

- *Analyse aspects of programs* - peer assessment prompts students to think about what aspects of their programs were wrong and what worked well when they compare their work with other students' work.

- *Evaluate whole programs* - making judgements about other students' work helps students to evaluate their own programs in a holistic manner. They are then better able to recognise what constitutes good and poor programs.
- *Synthesize better programs* – analysing and evaluating the different styles of solving programming problems helps students to create better programs.

We have developed web-based peer assessment (Sitthiworachart & Joy, 2003) with an Anonymous Communication Device (ACD) to support independent learning and encourage interaction with others, which is a key element in fostering deep learning (Biggs, 2001). This web-based peer assessment system has advantages over ordinary peer assessment because students can be more critical and feel free to express their own ideas about other students' work due to the anonymity the system provides (Wen et al., 2003). Using the ACD, students can discuss online and/or leave offline messages, the processing of the marks is automated, and the lecturer can easily monitor the marking and conversation. We describe the tool and the peer assessment process it supports, and report the students' reflection in marking programming. We evaluate the students' marking in deploying our web-based peer assessment system on a large computer programming course to justify the claim that our tool does encourage students to develop higher level cognitive skills in learning computer programming.

Web-Based Peer Assessment

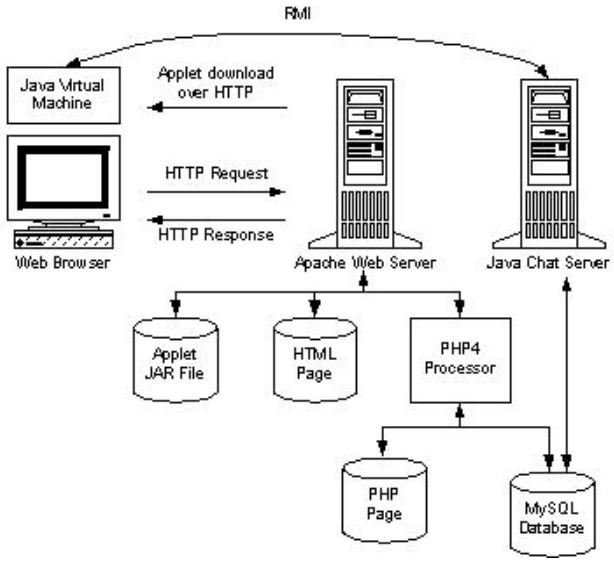

Figure 11 Architecture of the web-based peer assessment system

Web-based peer assessment is implemented using an Apache server enabled with PHP4 and accessed using a client Java applet for the ACD (see Figure 11). Students' responses to the questions forming the marking criteria were recorded in a MySQL database and were termed a "script". The ACD program itself is downloaded via http. Once the program has been downloaded, the applet creates its own connection to another server program, the chat server. The peer assessment investigation was performed on 213 first year undergraduate students enrolled on a first year UNIX programming module in the Computer Science department. During the process the students mark and provide feedback on 3 consecutive assignments, and each assignment is marked by an anonymised group of 3 students, using the web-based peer assessment system and the anonymous communication tool. Segers and Dochy (2001) reported that three was the appropriate number of students on which to base the mark for the peer assessment. These assignments are also independently double-marked by two module tutors, to provide an expert reference against which the marks awarded through the peer assessment process can be compared.

Process

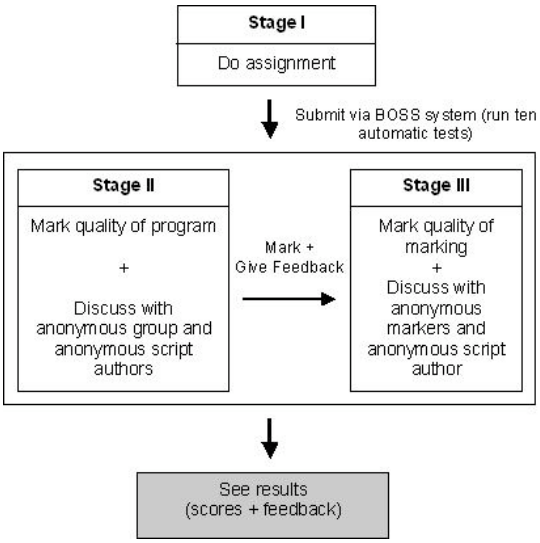

Figure 12 Peer assessment process

This peer assessment exercise was divided into three separate stages, as shown in Figure 12. The BOSS online submission system (Luck and Joy, 1999) used in the peer assessment process is capable of running automated tests, running students' submissions against a set of test cases constructed by the assessment designer. The automated test results were made available to markers along with the original submission through stages 2 and 3, in the hope that the extra information would help students to understand the submissions whilst marking. In the second stage, students mark the quality of programming; in the third stage, students mark the quality of marking, by responding to the ten questions that frame the marking criteria (see Table 2) and throughout the process discuss anonymously via the ACD. Thus students have reflected on their own ideas by providing the comment and discussing using the ACD in a variety of roles (script authors, marker, and feedback marker). When marking, students review and compare their own work with their peers' work. They analyse and evaluate other work, which leads to the development of evaluation and self-evaluation skills (Fallows & Chandramohan, 2001). When providing feedback, students correct other work and explain their arguments, which encourage the students to reflect. It can be of great value to the teacher both for teaching and for assessing course outcomes (McGourty et al., 1998).

Marking Criteria

Table 2 Marking criteria in the Unix programming module

READABILITY	1. The number of comments is	low	○○○○○	high
	2. Comments are	unhelpful	○○○○○	helpful
	3. The code is indented	inappropriately	○○○○○	appropriately
	4. Identifier names are	inappropriate	○○○○○	appropriate
CORRECTNESS	5. The program meets the specification	not at all	○○○○○	completely
	6. The code handles errors	inappropriate	○○○○○	appropriate
	7. The program finishes with an appropriate exit status	never	○○○○○	always
STYLE	8. The utilities have been selected	inappropriate	○○○○○	appropriate
	9. The program is structured	poorly	○○○○○	well
	10. Overall, following what the program is doing is	hard	○○○○○	easy

Orsmond et al. (2002) reported that students seemed unenthusiastic about creating marking criteria themselves. Therefore, in this peer assessment process, the marking criteria were set by the teacher. Marking criteria should be clear and marking scale should offer appropriate choices for marking (a 5-point Likert scale is recommended) in order to help student in making an accurate and fair judgment (Miller, 2003). Using specific marking criteria also helps students to understand what is expected of a good program and provokes deeper thinking about their own work (Smith et al., 2002). Students mark and provide feedback on other students' assignments by answering ten questions about:

- *Readability:* readability means that the program is easy to understand. Readability is helped by appropriate commenting - the comments should describe obscure code and algorithmic elements rather than the obvious. The layout of the code should reflect its intrinsic block structuring, and there are various accepted styles of code indentation. Readability can also be improved by a good choice of variable and method names (identifiers).
- *Correctness:* a program is correct if it does what it is supposed to. A program can be syntactically correct in that it compiles and runs, but be semantically incorrect. A correct program should handle errors, so that it does not fail when it is confronted with illegal values. For example, the program will anticipate values that might cause exceptions, perhaps by checking pre-conditions. A shell script should also finish with the correct exit status to indicate whether an error has occurred.
- *Style:* the first step to writing a program in a good style is selecting appropriate utilities so as to simplify the code. If students select inappropriate utilities their programs may be more complex than necessary, as students may have to program functionality that exists in another tool. The next step is to break the program down into logical sections, making a simple and concise program to solve a problem in a clear and complete manner. A well structured program with be more easily understood by the assessor.

Students' Reflection

Students analyse and evaluate the quality of program by providing comments and answering the ten questions about readability, correctness and style of program set out in Table 1. Students follow the suggested marking criteria when writing their subsequent assignments (i.e. including good indentation, helpful comments and clear structure). The following are the students' reflection on each marking criteria category.

Readability: Students think that a lack of comments, inconsistent indentation, and inappropriate identifier names make it difficult to follow the structure, even though the abstract structure of the program may be good. We have summarized the students' opinion about readability as follows:

- The authors should write introductory comments describing briefly what the code does and how it performs.
- Having more white space between the comments and code, and separating them properly by consistently using indentation helps to improve the presentation and make the code easily readable by others.
- Having comments on the end of the line has some disadvantages.
- Capitalizing the first letters of identifier names makes them a lot easier to read.

A sample student's comments on identifier names, and on an indentation problem in "if" statements is given below. They should be self-explanatory.

> ***Identifier names*** *are inappropriate. Authors have made use of names like "T", "L" etc. which I think is not a good idea because I think if we use these kind of names it will be really confusing if we have a long script and we have made use of lot of Identifier names. So may I suggest to the author that ……...*

> *The severe* ***indentation*** *caused by many if statements within other if statements made the indentation inappropriate. Completing one if statement at a time would have been better, allowing for more appropriate indentation.*

Correctness: Even though a program passes all the automatic tests, there may still be cases which the program will not handle correctly. Many students read a program carefully to find out whether the program meets the specification or not, the code handles errors appropriately, and the program finishes with an appropriate exit status.

> *The first 'if' statement outputs an error condition if the command is not run with one argument. This is* ***semantically incorrect*** *because the program …….. A possible solution would ……*

> *The* ***exit status*** *for all the automatic tests is 2. This is because there are syntax errors when the script is run meaning that it does not exit with the correct exit status. Therefore it never exits properly.*

> *The program is elegant, simple and concise but unfortunately it* ***does not entirely meet the specification****. The use of pattern matching would have enabled the problem to be solved in a complete manner. For example, instead of using the expression ……., the following expression could be used: ……...*

Style: Students suggest that the author should try to break down the problem into parts and organise them for solution in order rather than at random. They give some suggestions on how to improve the quality of program by changing the utilities, improving the solution style.

> *You use a good range of* ***utilities****, and your code is somewhat well structured. One small change you could make would be replace …….. by ……..., as it serves the same purpose but in much shorter and concise code.*

> *using "[" instead of "test" looks a bit nicer but of course "test" is ok. You started out using [e.g. "if [$# = 1]" but then swapped to" test" e.g. "test -d /dcs/03/$USER", maybe it's better to stick to one or the other and "[" makes the* ***structure*** *more clear in my opinion.*

Evaluation of Students' Marking

Table 3 Number of markings on the different aspects of comment in peer assessment

Comment	Assignment 1 (%)	Assignment 2 (%)	Assignment 3 (%)
students mainly gave comment and offered solutions	13.79	17.24	22.76
students mainly gave comment without offered solutions	22.76	20.00	13.79
students mainly made some positive comments	25.52	24.83	22.07
students mainly made some negative comments	12.41	6.90	8.28
students mainly made some positive & negative comments	3.45	6.90	4.83
students read the code carefully and analyse the problem	6.90	9.66	14.48
students did not read the code carefully, just rely on automatic test results	4.14	4.14	3.45
students find out the program they marked are better than their own work	1.38	2.07	4.14
students find out the program they marked are difficult to comment	2.07	2.07	1.38
students mainly made the comments on the programming syntax	2.07	2.76	2.07
students did not give comment or give short comments	5.52	3.45	2.76

A preliminary analysis of the sample marking in Table 3 indicates that students initially provided comments by analysing only programming problems without offering solutions (22.76% in assignment 1 and decreasing to 20%, and 13.79% in the subsequent assignments). However, they provided more solutions (from 13.79% to 22.76%), and read the code more carefully (from 6.90% to 14.48). They gave many positive comments (almost the highest percentage in every assignment), including compliments on the quality of the programs presented, the choice of utilities, the adherence to the assignment specification, and the comments in the programs. A few students found the programs they marked difficult to comment on because they did not understand some utilities or the programs they marked were not readable. Since students did not have marking experience initially, they hesitated to comment on other students' work, but students were less inhibited about commenting on the programs, offering solutions, and explaining their arguments after the first peer assessment in assignment 1. Finally, the number of students who did not give comment or give short comment was decreased in the subsequent assignments. This positive result indicates that peer assessment encourages students to reflect on their own ideas.

Consistency of Marking

Table 4 Standard deviation of mark from peers

Assignment	Standard Deviation (%)		
	Low <= 10	Medium >10	High >20
Assignment 1 (n = 205 students)	65.37	30.24	4.39
Assignment 2 (n = 212 students)	75.47	22.64	1.89
Assignment 3 (n = 213 students)	74.65	22.06	3.29

Each script was marked by three students, and we calculate the standard deviation of the three markers in order to know how spread out the marks are. If the standard deviation is less than a preset value, it is acceptable, but if the standard deviation is more than a given upper limit, it means that the marks from the three markers have a very wide range, and that the tutor may have to reconsider the marks for that student. The results from Table 4 show that fewer than 5% of students from all assignments received very wide ranging marks, and that the SD decreased significantly after the first assignment. We can conclude that most of the marking by students is consistent, as a high percentage of students show low standard deviation from all assignments. There is also evidence of consistency in the opinions of students marking the same assignment:

> *Marker1: there is no comment at all, that leads to a big problem of readability. Identifier names are nearly inappropriate.*
>
> *Marker2: This code had no comments. There is indentation, but there are some places where extra indentation would make it easier to read. The identifier names were not very clear.*
>
> *Marker3: No comments at all. Difficult to give any suggestions on commenting. The identifier "per" is not so clear to mean the list results. It's better to use 'list', or 'check' and so on.*

Each student's mark was compared against the mark awarded independently by the tutor. The marks in the last assignment awarded by the students were mostly higher than those awarded by the tutor, the means differing by approximately 8%, and scaling of peers' marks yielded results which, in almost all cases, matched the tutor's.

Students' Marking

Some students believe that only the expert or tutor can give them good comments and fair marking. However, marking of assignments by students is possible since they are given guidance, automatic test scores and results, a marking scheme, and well-explained marking criteria. Students can discuss anonymously with their group who mark the same script and ask the script author questions about the program via the ACD. Therefore all of these facilities help students in marking and better understanding of the program. The following are various suggestions on the same script from students with a range of different ability:

> *Marker1: The program passes all Automatic Tests but there are cases which the program will not handle correctly. For example, line 19 …….. The same applies to the test statements in lines 15 and 23. A possible solution to this would be ……...*
>
> *Marker2: The code is correct, since it passed all tests. through, there is a problem , as mentioned in "things to consider" The number of arguments can be checked ……*
>
> *Marker3: Simple utilities have been selected appropriately and the program is no more complex than it needs to be. The program also has a good sequential structure as it …….. This makes it very easy to see what is going on in the program, but ……...*

Moreover, there is evidence that involvement in the peer assessment exercise helps students to develop skills in critiquing programs. The following examples illustrate how the quality of feedback given by a student improved with each assignment. The student commented on the "Style" of

program, giving only comments without offering suggestions in the first assignment, analysing the program in the second assignment and finally providing some suggestions to improve the quality of the program in the last assignment. For this student, over the course of the assignments, the quality of program mark increased by 33% and the quality of feedback mark increased by 66%.

Assignment 1: The structure of the code is quite good. It is fairly easy to follow it.

Assignment 2: The program is simple and easy to follow. This person also wrote a function to test …….. I can see a very good use of functions …… .Just some small problems, ……...

Assignment 3: The author used a range of utilities... The easiest way to complete this script is to start from the bottom where …….. and then ……..It would be better that using …….... In general ,the utilities used in the script……..

Quality of Feedback

Table 5 Quality of feedback mark

	Assignment 1 (%)	Assignment 2 (%)	Assignment 3 (%)
Strongly Agree	46.84	47.40	50.84
Agree	26.65	30.28	29.79
Neutral	12.85	12.40	12.03
Disagree	7.47	6.91	5.92
Strongly Disagree	6.20	3.01	1.42

Students marked the quality of feedback by indicating how far they agreed that the suggestions that markers in previous step gave were relevant, well explained and useful to students (see Table 5). It appears that, when evaluating programs, the students were able to make more refined analyses after they had been through the peer assessment process. The results (Table 5) suggest that the quality and consistency of marking is increased, as the number of students who strongly agree increased from 46.84% to 50.84%, and the number of students who strongly disagree decreased from 6.20% to 1.42%. Most students agreed with the marking and the comments given by markers, but they also liked to add more critical suggestion of their own, spotting missing facts, giving alternative solutions, etc. However a few of the students who disagreed criticised the marks given as too generous or harsh, and identified comments as wrong, unhelpful, or inadequate. We have summarized the students' marking from the sample of marking on the quality of feedback below:

- 21 % of students agreed with the further suggestions.
- 32 % of students agreed and complimented the marking.
- 19 % of students disagreed but gave further suggestions.
- 0 % of students disagreed without giving further suggestions.
- 4 % of students agreed with mark but disagreed with the comments.
- 0.60 % of students disagreed with mark but agreed with the comments.
- 21 % of students agreed with some comments but disagreed with others. Explicitly:
    - 11 % of students thought some comments were not clear;
    - 20 % of students thought some comments were unhelpful;
    - 20 % of students thought markers had failed to understand some points; and
    - 49 % of students thought marker did not spot some points.
- 3 % of students made no comment on the marking.

Results & Discussions

Table 6 The average marks of students on the marking criteria

Marking Criteria	Mark (%)		
	Assignment 1	Assignment 2	Assignment 3
Readability	72.89	83.00	83.80
Correctness	84.57	81.58	82.55
Style	78.69	79.58	79.74

Table 6 presents the average marks of each marking criteria category from three assignments. The results indicate that, for assignment 1, students show programming strength in correctness (meeting specification, appropriate error handling, correct exit status) and programming weakness in readability (comment, indentation, identifier name). The students' weakness in readability is decreased by 14.97% in assignment 3, after they have been through the peer assessment process. This suggests that students pay more attention to the program readability after they have played the marker role in peer assessment. In effect, they realize how difficult it is to understand the program which has poor readability. However the quality of correctness is decreased, possibly because some students misunderstand the questions in the correctness marking criteria section in assignment 1. Both the readability and the correctness of programs improve from assignment 2 to assignment 3.

Table 7 Average mark from three assignments

	Assignment 1 (%)	Assignment 2 (%)	Assignment 3 (%)
Average mark of correctness of program	91.25	94.87	94.51
Average mark of quality of program	77.04	82.88	84.57
Average mark of quality of feedback	70.33	74.97	77.06

Table 7 shows the marks of each assignment, consisting of 50% from automatic test and 50% from peer assessment (30% for quality of program, 20% for quality of feedback). The percentage of most marks increases in the course of the assignments, with the exception of the program correctness mark in assignment 3, which is the most difficult assignment. However there is evidence that students can write better programs in the later assignments and that they also develop their analysis and evaluation skills (as the average mark of quality of feedback increase from 70.33% to 77.06%). Pearson correlation values indicated that quality of program and quality of feedback marks are significantly interrelated. A correlation between quality of feedback marks in assignment 1 and quality of program marks in assignment 2 is 0.397. Although a correlation of 0.397 is not high, it increases to a correlation of 0.484 in the next assignment. This is a significant result at the 0.01 level. The high positive correlation of 0.484 shows quality of program and quality of feedback marks are significantly interrelated. A mark on quality of feedback increases so the corresponding mark on quality of program in the subsequent assignments does the same. Therefore those students who can provide good critical comments on programs can also write programs of better quality. This is in keeping with previous findings that the more marking students did, the better their own results became (Bhalerao & Ward, 2001).

Conclusions

We have evaluated students' marking and described a peer assessment process, together with supporting web-based software, which we have used to test the effectiveness of peer assessment in learning programming languages. The process we have used is novel, since students are engaged not only in marking each other's work, but also in evaluating the quality of marking of their peers. Students do not only provide critical comments on other students' work, they also provide constructive

suggestions and propose solutions both in marking the quality of programs and the quality of marking. We can conclude that web-based peer assessment encourages students to develop their skills in analysis and evaluation in the context of programming. They also realise their own strengths and weaknesses when marking a good or poor piece of work. Preliminary evaluation of the exercise indicates that it has contributed positively to the students' learning experience.

Reference List

Bhalerao, A., & Ward, A. (2001). Towards electronically assisted peer assessment: a case study, *Association for Learning Technology journal (ALT-J)*, *9*(1), 26-37.

Biggs, J. (2001) *Asessing for Quality in Learning In: Assessment to Promote Deep learning*. American Association for Higher Education (AAHE), Washington, DC, 70-73.

Bloom, B.S. (1956) *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain*. Longmans, New York.

Brown, G., Bull, J., and Pendlebury, M. (1997), Assessing student learning in higher education, Routledge, London, 170-184.

Dochy, F., McDowell, L. (1997), Assessments as a tool for learning, *Studies in Educational Evaluation*, *23*(4), 279-298.

Falchikov, N. (2001), Learning Together: Peer tutoring in higher education, Routledge Falmer, London.

Fallows, S., & Chandramohan, B. (2001). Multiple Approaches to Assessment: reflections on use of tutor, peer and self-assessment. *Teaching in Higher Education*, *6*(2), 229-246.

Luck, M., & Joy, M. (1999). A Secure On-line Submission System. *Software - Practice and Experience*, *29*(8), 721-740.

McGourty, J., Dominick, P., & Reilly, R. (1998). Incorporating Student Peer Review and Feedback into the Assessment Process, (pp. 14-18). Proceedings of the 1998 FIE Conference.

Miller, P. (2003). The Effect of Scoring Criteria Specificity on Peer and Self-assessment. *Assessment & Evaluation in Higher Education*, *28*(4), 383-394.

Orsmond, P., Merry, S., & Reiling, K. (2002). The use of Exemplars and Formative Feedback when Using Student Derived Marking Criteria in Peer and Self-assessment. *Assessment & Evaluation in Higher Education*, *27*(4), 309-323.

Seger, M., & Dochy, F. (2001). New Assessment Forms in Problem-based Learning: the value-added of the students' perspective. *Studies in Higher Education, 26*(3), 327-343.

Sitthiworachart, J., & Joy, M. (2003). Web-based Peer Assessment in Learning Computer Programming. Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies: ICALT03, Athens, Greece.

Sluijsmans, D., Dochy, F., & Moerkerke, G. (1999). Creating a Learning Environment by Using Self-Peer-and Co-Assessment. *Learning Environments Research, 1*, 293-319.

Smith, H., Cooper, A., & Lancaster, L. (2002). Improving the Quality of Undergraduate Peer Assessment: A Case for Student and Staff Development. *Innovations in Education and Teaching International: IETI, 39*(1), 71-87.

Somervell, H. (1993), Issues in assessment, enterprise and higher education: the case for self-, peer and collaborative assessment, *Assessment and Evaluation in Higher Education*, *18*, 221-233.

Topping, K., Smith, E., Swanson, I., & Elliot, A. (2000). Formative Peer Assessment of Academic Writing Between Postgraduate Students. *Assessment & Evaluation in Higher Education*, *25*(2), 149-169.

Wen, M., Tsai, C., & Chang, C. (2003). Attitudes toward Peer Assessment: Perspectives from College Students and Inservice Teachers. In Y. Chee, N. Law, K. Lee & D. Suthers (Eds.), *The "Second Wave" of ICT in Education: from Facilitating Teaching and Learning to Engendering Education Reform*, (pp. 181-184). Proceedings of the International Conference on Computers in Education 2003: ICCE2003, Hong Kong.

Brief author details

Jirarat Sitthiworachart: Jirarat is currently studying a PhD in Computer Science at the University of Warwick, UK. Her research area is the use of novel software tools for enhancing students' higher cognitive skills in computer science.

Mike Joy: Dr Joy is a Senior Lecturer in Computer Science at the University of Warwick. His current research interests include educational technology and agent-based systems, and recent work has centred around automatic assessment and plagiarism detection.